Theoretical Computer Science 411 (2010) 3614–3624

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Provable secure identity based generalized signcryption scheme

Gang Yu [a,*], Xiaoxiao Ma [b], Yong Shen [a], Wenbao Han [a]

[a] Department of Applied Mathematics, Zhengzhou Information Science and Technology Institute, Zhengzhou 450002, China
[b] School of Surveying and Land Information Engineering of Henan Polytechnic University, Jiaozuo 454003, China

ARTICLE INFO

Article history:
Received 18 November 2009
Received in revised form 27 May 2010
Accepted 6 June 2010
Communicated by X. Deng

Keywords:
Generalized signcryption
Signature
Encryption
Bilinear pairing
Identity based cryptography

ABSTRACT

According to actual needs, a generalized signcryption scheme can flexibly work as an encryption scheme, a signature scheme or a signcryption scheme. In this paper, firstly, we give a security model for identity based generalized signcryption which is more complete than the existing model. Secondly, we propose an identity based generalized signcryption scheme. Thirdly, we give the security proof of the new scheme in this complete model. Compared with existing identity based generalized signcryption, the new scheme has less implementation complexity. Moreover, the new scheme has comparable computation complexity with the existing normal signcryption schemes.

1. Introduction

Encryption and signature are fundamental tools of Public Key Cryptography for confidentiality and authenticity respectively. Traditionally, these two main building-blocks have been considered as independent entities. However, these two basic cryptographic techniques may be combined together in various ways, such as sign-then-encrypt and encrypt-then-sign, in many applications to ensure privacy and authenticity simultaneously. To enhance efficiency, Zheng [17] proposed a novel conception named signcryption, which can fulfill both the functions of signature and encryption in a logical step. Compared with traditional methods, signcryption has less computation complexity, less communication complexity and less implementation complexity. As a result of the signcryption scheme having so many advantages and extensive application prospects, many public key based signcryption schemes have been proposed [18,2,10,8].

Identity-based cryptography was introduced by Shamir [15] in 1984, in which the public keys of users are respectively their identities and the secret keys of users are created by a credit third party named Public Key Generator (PKG). In this way, the identity-based cryptography greatly relieves the burden of public key management and provides a more convenient alternative to the conventional public key infrastructure. In [15], Shamir proposed an identity based signature scheme but for many years there was not an identity based encryption scheme. In 2001, Boneh and Franklin [1] using bilinear pairing gave a practical secure identity based encryption scheme. The first identity based signcryption scheme was proposed by Malone-Lee [13] along with a security model. Since then, many identity based signcryption schemes have been proposed [12,3,7,5].

Signcryption has considered these application environments that need simultaneous message privacy and data integrity. However, in some applications, these two properties are not essential. That is, sometimes only message confidentiality is needed or sometimes only authenticity is needed. In this case, in order to ensure privacy or authenticity separately, signcryption must preserve the sign module or encryption module, which must increase the corresponding computation complexity

* Corresponding author. Tel.: +86 13783717183.
E-mail address: ygygang@126.com (G. Yu).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.06.003

and implementation complexity. To decrease implementation complexity, Han et al. [9] proposed a new primitive called generalized signcryption, which can work as an encryption scheme, a signature scheme or a signcryption scheme, and gave a generalized signcryption based on ECDSA. Wang et al. [16] gave the formal security notions for this new primitive and improved the original generalized signcryption proposed by Han et al. [9]. In [16], Wang et al. pointed out some open problems. One of these problems is to enhance efficiency. Another of these problems is to design identity based generalized signcryption scheme.

Lal et al. [11] gave an identity based generalized signcryption scheme (IDGSC). However, after much study, we find his security model is not complete. And his scheme is not secure under the complete security model for IDGSC. In this paper, our main works include three aspects. Firstly, in the second section, we give the definition of IDGSC and the security model for IDGSC. Secondly, in Section 3, we propose an efficient IDGSC. Thirdly, in Section 4, we give the efficiency analysis and security results.

2. IDGSC and its security notions

2.1. Definition of IDGSC

Firstly, we will review the algorithm constitution of identity based encryption (IDEC), identity based signature (IDSG) and identity based signcryption (IDSC). Then, we will introduce the algorithms that consist of an identity based generalized signcryption (IDGSC).

Definition 1. A normal identity based encryption scheme

$$IDEC = (Setup, Extract, Encrypt, Decrypt)$$

consists of four algorithms.
Setup: This is the system initialization algorithm. On input of the security parameter 1^k, this algorithm generates the system parameters *params* and the PKG generates his master key s and public key P_{Pub}. The global public parameters include *params* and P_{Pub}. We write $((params, P_{Pub}), s) \leftarrow Setup(1^k)$.
Extract: This is the user key generation algorithm. Given some user's identity ID, PKG uses it to produce a pair of corresponding public/private keys. We write $(S_{ID}, Q_{ID}) \leftarrow Extract(ID, s)$.
Encrypt: It takes as input a receiver's identity ID_r and a message m, using the public parameters $(params, P_{Pub})$, outputs a ciphertext ε. We write $\varepsilon \leftarrow Encrypt(ID_r, m)$.
Decrypt: It takes as input a receiver's private key S_r and a ciphertext ε, using the public parameters $(params, P_{Pub})$, outputs a message m or the invalid symbol $\perp$. We write $m \leftarrow Decrypt(S_r, \varepsilon)$.

Definition 2. A normal identity based signature scheme

$$IDSG = (Setup, Extract, Sign, Verify)$$

consists of four algorithms.
Setup: It is the same as the corresponding Setup algorithm in Definition 1.
Extract: It is the same as the Extract algorithm in Definition 1.
Sign: This algorithm takes as input a signer's private key S_s and a message m, using the public parameters $(params, P_{Pub})$, outputs a signature σ. We write $\sigma \leftarrow Sign(S_s, m)$.
Verify: This algorithm takes as input the signer's public key Q_s, a message m and the corresponding signature σ, and outputs the valid symbol $\top$ or the invalid symbol $\perp$. We write $(\top \ or \ \perp) \leftarrow Verify(Q_s, m, \sigma)$.

Definition 3. A normal identity based signcryption scheme

$$IDSC = (Setup, Extract, Signcrypt, Unsigncrypt)$$

consists of four algorithms.
Setup: It is the same as the corresponding Setup algorithm in Definition 1.
Extract: It is the same as the Extract algorithm in Definition 1.
Signcrypt: This algorithm takes as input the sender's private key S_s, the receiver's public key Q_r and a message m, using the public parameters $(params, P_{Pub})$, outputs a ciphertext δ. We write $\delta \leftarrow SC(S_s, Q_r, m)$.
Unsigncrypt: This algorithm takes as input the sender's public key Q_s, the receiver's secret key S_r and a ciphertext δ, using the public parameters $(params, P_{Pub})$, outputs a message m or the invalid symbol $\perp$. We write $m \leftarrow UC(Q_s, S_r, \delta)$.

The generalized signcryption scheme can work as encryption scheme, signature scheme and signcryption scheme according to different needs. Let $IDSG = (Setup, Extract, Sign, Verify)$, $IDEC = (Setup, Extract, Encryp\,t, Decrypt)$ and $IDSC = (Setup, Extract, Signcrypt, Unsigncrypt)$ respectively be an identity based signature scheme, encryption scheme and signcryption scheme.

Definition 4. An identity based generalized signcryption scheme $IDGSC = (Setup, Extract, GSC, GUC)$consists of following four algorithms:
Setup: It is the same as the corresponding Setup algorithm in Definition 1.
Extract: It is the same as the Extract algorithm in Definition 1.
GSC: for a message m,
-When $ID_s \in \Phi(ID_s = 0), \varepsilon \leftarrow GSC(\Phi, Q_r, m) = Encrypt(Q_r, m)$.
-When $ID_r \in \Phi(ID_r = 0), \sigma \leftarrow GSC(S_s, \Phi, m) = Sign(S_s, m)$.
-When $ID_s \notin \Phi, ID_r \notin \Phi, \delta \leftarrow GSC(S_s, Q_r, m) = SC(S_s, Q_r, m)$.
GUC: to unsigncrypt a ciphertext δ,
-When $ID_s \in \Phi(ID_s = 0), m \leftarrow GUC(\Phi, Q_r, \varepsilon) = Decrypt(Q_r, \varepsilon)$.
-When $ID_r \in \Phi(ID_r = 0), (\top, \perp) \leftarrow GUC(S_s, \Phi, \sigma) = Verify(S_s, \sigma)$.
-When $ID_s \notin \Phi, ID_r \notin \Phi, m \leftarrow GUC(Q_s, S_r, \delta) = UC(Q_s, S_r, \delta)$.

2.2. Security models for IDGSC

In our security model, there are seven types of query that the adversary A may issue to the challenger C for answers. In the following text, "$Alice\{Text1\} \rightarrow Bob$, and then $Bob\{Text2\} \rightarrow Alice$" denotes that Alice submits Text1 to Bob, and then Bob responds with Text2 to Alice.
Extract query: $A\{ID\} \rightarrow C$, and then $C\{S_{ID} = Extract(ID)\} \rightarrow A$
Sign query: $A\{ID_s, m\} \rightarrow C$, and then $C\{\sigma = Sign(S_s, m)\} \rightarrow A$
Verify query: $A\{ID_s, \sigma\} \rightarrow C$, and then $C\{(\top \text{ or } \perp) = Verify(Q_s, \sigma)\} \rightarrow A$
Encrypt query: $A\{ID_r, m\} \rightarrow C$, and then $C\{\varepsilon = Encrypt(Q_r, m)\} \rightarrow A$
Decrypt query: $A\{ID_r, \varepsilon\} \rightarrow C$, and then $C\{m = Decrypt(S_r, \varepsilon)\} \rightarrow A$
GSC query: $A\{ID_s, ID_r, m\} \rightarrow C$, and then $C\{\delta = GSC(S_s, Q_r, m)\} \rightarrow A$
GUC query: $A\{ID_s, ID_r, \delta\} \rightarrow C$, and then $C\{m = GUC(Q_s, S_r, \delta)\} \rightarrow A$

The generalized signcryption can work in three modes: in signature mode, in encryption mode and in signcryption mode, denoted IDGSC-IN-SG, IDGSC-IN-EN and IDGSC-IN-SC respectively. Firstly, we define the confidentiality of IDGSC-IN-EN (Definition 5) and IDGSC-IN-SC (Definition 6) separately.

Definition 5. IND-(IDGSC-IN-EN)-CCA Security
Consider the following game played by a challenger C and an adversary A.
Game 1
Initialize. Challenger C runs $Setup(1^k)$ and sends the public parameters $(params, P_{Pub})$ to the adversary A. C keeps master key s secret.
Phase 1. In Phase 1, A performs a polynomially bounded number of above seven types of queries. These queries made by A are adaptive; that is every query may depend on the answers to previous queries.
Challenge. The adversary A chooses two identities $ID_A = 0, ID_B \neq 0$ and two messages m_0, m_1. Here, the adversary A cannot have asked Extract query on ID_B in Phase 1. The challenger C flips a fair binary coin γ, encrypts m_γ and then sends the target ciphertext ε^* to A.
Phase 2. In this phase, A again asks a polynomially bounded number of the above queries with a natural restriction that he cannot make Extract queries on ID_B, and he cannot ask Decrypt query on target ciphertext ε^*.
Guess. Finally, A produces his guess γ' on γ, and wins the game if $\gamma' = \gamma$.

A's advantage of winning Game 1 is defined to be $Adv^{ind-cca2}_{A^{idgsc-in-en}}(t, p) = |2P[\gamma' = \gamma] - 1|$. We say that identity based generalized signcryption in encryption mode is IND-(IDGSC-IN-EN)-CCA secure if no polynomially bounded adversary A has a non-negligible advantage in Game 1.

Definition 6. IND-(IDGSC-IN-SC)-CCA Security
Consider the following game played by a challenger C and an adversary A.
Game 2
Initialize. and *Phase* 1.
Challenger C and adversary A act the same as they do in the corresponding stage in Game 1.
Challenge. The adversary A chooses two identities $ID_A \neq 0, ID_B \neq 0$ and two messages m_0, m_1. Here, the adversary A cannot have asked Extract query on ID_B in Phase 1. The challenger C flips a fair binary coin γ, signcrypts m_γ and then sends the target ciphertext δ^* to A.
Phase 2. In this phase, A asks a polynomially bounded number of above queries just with a natural restriction that he cannot make Extract queries on ID_B, and he cannot ask Unsigncrypt query on target ciphertext δ^*.
Guess. Finally, A produces his guess γ' on γ, and wins the game if $\gamma' = \gamma$.

A's advantage of winning Game 1 is defined to be$Adv^{ind-cca2}_{A^{idgsc-in-sc}}(t, p) = |2P[\gamma' = \gamma] - 1|$. We say that identity based generalized signcryption in signcryption mode is IND-(IDGSC-IN-SC)-CCA secure if no polynomially bounded adversary A has a non-negligible advantage in Game 2.

Note 1. The differences between Definitions 5 and 6 deserve to be mentioned. Firstly, in Phase 2 of Definition 5, the adversary is prohibited from making Decrypt query on the challenge ciphertext. However, he can transform the challenge ciphertext into some valid signcryption ciphertext and make Unsigncrypt query on the corresponding signcryption ciphertext. Secondly, the adversary is restricted not to make Unsigncrypt query on the challenge ciphertext in Phase 2 of Definition 6. But, he can transform the challenge ciphertext into some valid encryption ciphertext and make Decrypt query on the corresponding encryption ciphertext. Such differences are not considered in the security model proposed by Lal et al. [11].

Secondly, we define the unforgeability of IDGSC-IN-SG (Definition 7) and IDGSC-IN-SC (Definition 8) separately.

Definition 7. EF-(IDGSC-IN-SG)-ACMA Security

Consider the following game played by a challenger C and an adversary A.

Game 3

Initialize. Challenger C runs $Setup(1^k)$ and sends the public parameters $(params, P_{Pub})$ to the adversary A. C keeps the master key s secret.

Probe. In this phase, A performs a polynomially bounded number of above seven kinds of queries.

Forge. Finally, A produces two identities ID_A, ID_B, where $ID_B = 0$, and a ciphertext $\sigma^* = (X^*, m^*, V^*)$. The adversary wins the game if: $ID_A \neq 0$; $Verify(m^*, ID_A, (X^*, V^*)) = \top$; no Extraction query was made on ID_A; (X^*, V^*) was not result from $Sign(m^*)$ query with signer ID_A.

We define the advantage of A to be $adv^{ef-acma}_{A^{idgsc-in-sg}}(t, p) = \Pr[Awins]$. We say that an identity based generalized signcryption in signature mode is EF-(IDGSC-IN-SG)-ACMA secure if no polynomially bounded adversary has a non-negligible advantage in Game 3.

Definition 8. EF-(IDGSC-IN-SC)-ACMA Security

Consider the following game played by a challenger C and an adversary A.

Game 4

Initialize. Challenger C runs $Setup(1^k)$ and sends the public parameters $(params, P_{Pub})$ to the adversary A. C keeps the master key s secret.

Probe. In this phase, A performs a polynomially bounded number of above seven kinds of queries.

Forge. Finally, A produces two identities ID_A, ID_B, and a ciphertext $\sigma^* = (X^*, C^*, V^*)$. Let m^* be the result of unsigncrypting δ^* under the secret key corresponding to ID_B.The adversary wins the game if: $ID_A \neq 0$; $ID_A \neq ID_B$; $Verify(m^*, ID_A, (X^*, V^*)) = \top$; no Extraction query was made on ID_A; (δ^*, ID_A, ID_B) was not output by a Signcrypt query.

We define the advantage of A to be $Adv^{ef-acma}_{A^{idgsc-insc}}(t, p) = \Pr[Awins]$. We say that an identity based signcryption in signcryption mode is EF-(IDGSC-IN-SC)-ACMA secure if no polynomially bounded adversary has a non-negligible advantage in Game 4.

Note 2. The differences between Definitions 7 and 8 also need to be noticed. In Definition 7, the forged signature is not obtained from the Sign query. But it can be transformed from some valid signcryption ciphertext that is gotten from Signcrypt query. In contrast, in Definition 8, the forged signcryption ciphertext is not the output of Signcrypt query. But it can be transformed from some answer of the Sign query. Such differences are not considered in the security model proposed by Lal et al. [11]. Consequently, in Lal et al. [11]'s scheme, the adversary can easily forge a valid signature through a corresponding Signcrypt query and Unsigncrypt query.

3. Our scheme

3.1. Description of our scheme

Before describing our scheme we need to define a special function $f(ID)$, where $ID \in \{0, 1\}^{n_1}$. If identity is vacant, that is $ID \in \Phi$, let $ID = 0$, $f(ID) = 0$; in other cases, $f(ID) = 1$. The concrete algorithms of our scheme are described as follows.

Setup: Given the security parameter 1^k, this algorithm outputs: two cycle groups $(G_1, +)$ and $(G_2, \cdot)$ of prime order q, a generator P of G_1, a bilinear map $\hat{e} : G_1 \times G_1 \to G_2$ between G_1 and G_2, four hash functions:

$$H_0 : \{0, 1\}^{n_1} \to G_1^*; \qquad H_1 : G_2 \to \{0, 1\}^{n_2} \times \{0, 1\}^{n_1} \times G_1^*;$$
$$H_2 : \{0, 1\}^{n_2} \times \{0, 1\}^{n_1} \times \{0, 1\}^{n_1} \to Z_q^*; \qquad H_3 : \{0, 1\}^{n_2} \times G_1 \to Z_q^*$$

where n_1 and n_2 respectively denote the bit length of user's identity and the message. Here H_0, H_1 needs to satisfy an additional property: $H_0(0) = \vartheta$, $H_1(1) = 0$, where ϑ denotes the infinite element in group G_1.The system parameters are $params = \{G_1, G_2, q, n_1, n_2, \hat{e}, P, H_0, H_1, H_2, H_3\}$. Then, PKG chooses s randomly from Z_q^* as his master key, and computes $P_{Pub} = sP$ as his public key. The global public parameters are $(params, P_{Pub}) = \{G_1, G_2, q, n_1, n_2, \hat{e}, P, P_{Pub}, H_0, H_1, H_2, H_3\}$.

Extract: each user in the system with identity ID_U; his public key $Q_U = H_0(ID_U)$ is a simple transform from his identity. Then PKG computes private key $S_U = sQ_U$ for ID_U.

Generalized Signcryption: Suppose Alice with identity ID_A wants to send message m to Bob whose identity is ID_B, he does as following:

- Computes $f(ID_A)$ and $f(ID_B)$.
- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||ID_A||ID_B)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2P + f(ID_A) \cdot h_3 \cdot S_A)$.
- Computes $Q_B = H_0(ID_B)$ and $w = \hat{e}(P_{Pub}, Q_B)^{r \cdot f(ID_B)}$.
- Computes $h_1 = H_1(w)$ and $y = m||ID_A||V \oplus h_1$.
- Sends (X, y) to Bob.

Generalized Unsigncryption: After receiving (X, y):

- Computes $f(ID_B)$.
- Computes $w = \hat{e}(X, S_B)^{f(ID_B)}$, $h_1 = H_1(w)$, $m||ID_A||V = y \oplus h_1$.
- Computes $h_2 = H_2(m||ID_A||ID_B)$ and $h_3 = H_3(m||X)$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_A)^{h_3 \cdot f(ID_A)}$, if not, returns $\perp$. Else, returns m.

3.2. Correctness

There are three cases to be considered.

Case 1. IDGSC-IN-SC

In this case, there is $ID_A, ID_B \notin \Phi$ (That is $ID_A, ID_B \neq 0$), so $f(ID_A) = f(ID_B) = 1$ and the scheme is actually a signcryption scheme. It is easy to verify that:

$w = \hat{e}(P_{Pub}, Q_B)^r = \hat{e}(X, S_B)$;

$\hat{e}(X, V) = \hat{e}(rP, r^{-1}(h_2P + h_3 \cdot S_A) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_A)^{h_3}$;

$UC(ID_A, ID_B, SC(ID_A, ID_B, m)) = m$.

So our scheme in signcryption mode is correct.

Case 2. IDGSC-IN-SG

In this case, there is $ID_A \notin \Phi$, $ID_B \in \Phi$ (That is $ID_A \neq 0$, $ID_B = 0$.), so $f(ID_A) = 1, f(ID_B) = 0$. The generalized signcryption scheme in signature mode is as follows:

Sign:

- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||ID_A||0)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2P + f(ID_A) \cdot h_3 \cdot S_A) = r^{-1}(h_2P + h_3 \cdot S_A)$.
- Computes $Q_B = H_0(0) = \vartheta$ and $w = \hat{e}(P_{Pub}, \vartheta)^{r \cdot f(ID_B)} = 1$.
- Computes $h_1 = H_1(w) = H_1(1) = 0$ and $y = m||ID_A||V \oplus 0 = m||ID_A||V$.
- Outputs the signature$(X, m||ID_A||V)$.

Verify:

- Computes $h_2 = H_2(m||ID_A||0)$ and $h_3 = H_3(m||X)$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_A)^{h_3}$, if not, returns $\perp$.

In fact, the reduced signature scheme is the signature scheme, denoted PSG, proposed by Paterson [14].

Case 3. IDGSC-IN-EN

In this case, there is $ID_A \in \Phi$, $ID_B \notin \Phi$ (That is $ID_A = 0$, $ID_B \neq 0$.), so $f(ID_A) = 0, f(ID_B) = 1$. The generalized signcryption scheme in encryption mode is as follows:

Encrypt:

- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||0||ID_B)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2P + f(ID_A) \cdot h_3 \cdot S_A) = r^{-1}h_2P$.
- Computes $Q_B = H_0(ID_B)$ and $w = \hat{e}(P_{Pub}, Q_B)^r$.
- Computes $h_1 = H_1(w)$ and $y = m||0||V \oplus h_1$.
- Sends (X, y) to Bob.

Decrypt:

- Computes $f(ID_B)$.
- Computes $w = \hat{e}(X, S_B)^{f(ID_B)} = \hat{e}(X, S_B)$ and $h_1 = H_1(w)$.
- Computes $m||0||V = y \oplus h_1$.
- Computes $h_2 = H_2(m||0||ID_B)$ and $h_3 = H_3(m||X)$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2}$. if not, returns $\perp$. Else, returns m.

Actually, the reduced encryption scheme is combination of the basic encryption scheme, denoted BFE, proposed by Boneh and Franklin [1] and a one-time signature scheme.

Table 1
Comparison between the dominant operations required for IDGSC and other schemes.

Schemes	Sign/Encrypt			Decrypt/Verify		
	mul. in G_1	exps. in G_2	$\hat{e}$ cps.	mul. in G_1	exps. in G_2	$\hat{e}$ cps.
[13]	3	0	0(+1)	0	1	3(+1)
[12]	2	2	0(+2)	0	2	2(+2)
[3]	3	1	0(+1)	2	0	3(+1)
[7]	2	0	0(+2)	1	0	4
[5]	3	0	0(+1)	1	0	3
[11]	5	0	0(+1)	1	0	3(+1)
NIDGSC	3	1	0(+1)	0	2	2(+2)

4. Efficiency analysis and security results

4.1. Efficiency analysis

The main purpose of generalized signcryption is to reduce implementation complexity. According to different application environments, generalized signcryption can fulfill the function of signature, encryption or signcryption respectively. However, the computation complexity may increase compared with the normal signcryption scheme. As with [9,16], these schemes all need an additional secure MAC function which not only increase the computation complexity but also the implementation complexity. Fortunately, these additional requirements are not needed in our scheme. Moreover, our scheme is as efficient as [5], which is the most efficient identity based signcryption scheme. In Table 1 below we compare the computation complexity of our scheme, denoted NIDGSC, with several famous signcryption schemes. We use mul., exps. and cps. as abbreviations for multiplications, exponentiations and computations respectively. Here, the computations that can be pre-calculated will be denoted by (+?).

4.2. Security results

In this section we will state the security results for our scheme under the security model defined in Section 2.2. Our results are all in the random oracle model. In each of the results below we assume that the adversary makes q_i queries to H_i for $i = 0, 1, 2, 3$. q_s and q_u denote the number of Signcrypt and Unsigncrypt queries made by the adversary respectively. n_3 and n_4 denote the bit length of an element in group G_1 and G_2 respectively.

Theorem 1. *If there is an EF-ACMA adversary A of NIDGSC in signature-mode that succeeds with advantage* $adv^{ef-acma}_{A^{idgsc-in-sg}}(t, p)$, *then there is a simulator C that can forge a valid signature of PSG with advantage* $\xi \approx adv^{ef-acma}_{A^{idgsc-in-sg}}(t, p)$.

When NIDGSC works as a signature scheme, it is actually the signature scheme, PSG, proposed by Paterson [14]. The PSG scheme itself is EF-ACMA secure. Considering Signcrypt/Unsigncrypt query that is absent in the normal signature scheme, these queries are useless to the adversary of NIDGSC-IN-SG. Because the identities of sender and receiver are included in the signature. There are two ways to modify these values. First, the adversary must find a special Hash collision. Second, the adversary succeeds in solving the ECDLP [6] problem. In such cases, the adversary has negligible advantage to modify these values. So an EF-ACMA adversary can attack the PSG scheme if he can attack NIDGSC in signature mode.

Theorem 2. *Let* $Adv^{ind-cca2}_{A^{idgsc-in-en}}(t, p) = \xi$ *be advantage of an IND-CCA2 adversary A of NIDGSC in encryption-mode, then* ξ *is polynomial time negligible.*

When NIDGSC works as an encryption scheme, it is actually the combination of the basic identity based encryption scheme proposed by [1] and a one-time signature scheme. Owing to the theorem proposed by Canetti et al. [4], this combined encryption scheme is secure against normal adaptive chosen-ciphertext attack. Considering Signcrypt/Unsigncrypt query, the adversary cannot transform the target encryption ciphertext into a valid signcryption ciphertext. This conclusion is based on the EF-ACMA security of PSG. So NIDGSC in encryption mode is IND-CCA2 secure.

Theorem 3. *If A can forge valid signcryption ciphertext of NIDGSC in signcryption-mode successfully with advantage* $Adv^{ef-acma}_{A^{idgsc-insc}}(t, p)$, *then there is a simulator C that can forge valid signature of PSG with advantage* ξ*:*

$$\xi \geqslant Adv^{ef-acma}_{A^{idgsc-insc}}(t, p) + (q_1 \cdot q_s)/2^{n_4} + q_u/(2^{n_2} \cdot 2^{n_1} \cdot 2^{n_3}).$$

The corresponding proofs are given in Appendix A.

Theorem 4. *If there is an IND-IBSC-CCA2 adversary A of NIDGSC in signcryption-mode that succeeds with advantage* $Adv^{ind-cca2}_{A^{idgsc-in-sc}}(t, p)$, *then there is a challenger C running in polynomial time that solves the weak BCDH problem with advantage* ξ*:*

$$\xi \geqslant Adv^{ind-cca2}_{A^{idgsc-in-sc}}(t, p)/(q_0 \cdot q_1).$$

The definition of weak BCDH problem and corresponding proofs are given in Appendix B.

5. Conclusions

In this paper, we define the security model for IDGSC and propose an efficient IDGSC which is proved secure under this security model. Compared with existing generalized signcryption schemes, our scheme does not need an extra secure MAC function. So it has less implementation complexity. What's more, it is almost as efficient as the normal signcryption scheme.

An interesting open question is to design a non-ID based (public key or Certificateless) generalized signcryption scheme that does not need an additional MAC function.

Acknowledgements

This work is supported by 863 Project of China (No. 2009AA01Z417). The authors would like to thanks the anonymous referees for their helpful comments.

Appendix A. Proof of Theorem 3

We will reduce the attack to EF-ACMA of NIDGSC to EF-ACMA of PSG proposed by Paterson [14]. Hence, we define two experiments Exp 1 and Exp 2. In each experiment, the private and public key and the Random Oracle's coin flipping space are not changed. The difference between Exp 1 and Exp 2 comes from rules of oracle service that challenger provides for the adversary.

Exp 1

In this experiment, we use the standard technique to simulate Hash functions used in our scheme. It is well-known that no adversary can distinguish between this environment and the real environment in polynomially bounded time. Let S_0 denote the event that EF-ACMA adversary can attack NIDGSC successfully in Exp 1.

The challenger C needs to keep four lists L_i, $i = 0, 1, 2, 3$ which are vacant at the very beginning. These lists are used to record answers to the corresponding Hash H_i, $i = 0, 1, 2, 3$ query.

Setup. At the beginning, the challenger C runs the algorithm $Setup(1^k)$ and acts as PKG. That is, he generates the global public system parameters ($params$, P_{Pub}) and the master private key s. Then, he sends ($params$, P_{Pub}) to the adversary A.

Probe. We now describe how the challenger simulates various queries.

Simulator: $H_0(ID_U)$

- If the record (ID_U, Q_U, S_U) is found in L_0, then returns Q_U.
- Else chooses Q_U randomly from G_1^*; computes $S_U = sQ_U$; stores (ID_U, Q_U, S_U) in L_0 and returns Q_U.

Simulator: $H_1(w)$

- Searches (w, h_1) in the list L_1. If such a pair is found, returns h_1.
- Otherwise chooses h_1 randomly from $\{0, 1\}^{n_2} \times \{0, 1\}^{n_1} \times G_1^*$, and puts (w, h_1) into L_1 and returns h_1.

Simulator: $H_2(m||ID_A||ID_B)$

- Searches $(m||ID_A||ID_B, h_2)$ in List L_2. If such a pair is found, returns h_2.
- Otherwise chooses h_2 randomly from Z_q^*, and puts $(m||ID_A||ID_B, h_2)$ into L_2 and returns h_2.

Simulator: $H_3(m||X)$

- Searches $(m||X, h_3)$ in the list L_3. If such a pair is found, returns h_3.
- Otherwise chooses h_3 randomly from Z_q^*, and puts $(m||X, h_2)$ into L_3 and returns h_3.

Simulator: $Extract(ID_U)$

We assume that A makes the query $H_0(ID_U)$ before it makes extract query for ID_U.

- Searches L_0 for the entry (ID_U, Q_U, S_U) corresponding to ID_U, and responds with S_U.

Simulator: $Sign(ID_A, m)$, $Verify(ID_B, \sigma)$

The challenger can easily answer these queries for the adversary. Because the challenger initializes the system and he knows the master key. So he can use signer ID_A's private key to sign message m and use the receiver ID_B's public key to verify the signature σ faithfully according to IDGSC-IN-SG. The only difference is substituting the above Hash simulators for Hash functions.

Simulator: $Encrypt(ID_B, m)$, $Decrypt(ID_B, \varepsilon)$

The challenger can get receiver ID_B's public key and private key. So he can supply these services for the adversary. Also the Hash functions in the scheme use the above Hash simulators.

Simulator: $GSC(ID_A, ID_B, m)$, $GUC(ID_A, ID_B, \delta)$

The challenger can get sender ID_A's public key and private key and receiver ID_B's public key and private key. So he can supply these services for the adversary. Here, the Hash functions also use the above Hash simulators.

Exp 2

In this experiment, we will remove the layer of encryption and reduce the signcryption scheme to PSG scheme. In the Setup phase, the challenger initializes the system just like he does in Exp 1. In the Probe phase, besides following simulators, challenger acts same with Exp 1.

Simulator: $Sign(ID_A, m)$, $Verify(ID_B, \sigma)$

Here, the challenger will follow PSG to accomplish these simulations.

Simulator: $GSC(ID_A, ID_B, m)$

Here, the challenger will keep another list L_s to record the GSC queries that the adversary asks.

- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||ID_A||ID_B)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2P + h_3 \cdot S_A)$.
- Selects h_1 uniformly from $\{0, 1\}^{n_2} \times \{0, 1\}^{n_1} \times G_1^*$ and adds $(*, h_1)$ in List L_1. The first element is vacant, and will be given some value later.
- Computes $y = m||ID_A||V \oplus h_1$ and adds (X, y, V, ID_A, ID_B, m) to List L_s.
- Outputs ciphertext (X, y). (Here, h_2, h_3 come from the corresponding Hash Simulators.)

Simulator: $GUC(ID_A, ID_B, \delta)$

- Searches $(*||ID_A||ID_B, *)$ in the list L_2, if such a record $(m||ID_A||ID_B, h_2)$ is found, goes to the next step. Else, returns $\perp$.
- Searches $(m||*, *)$ in the list L_3, if such a record $(m||X, h_3)$ is found, goes to the next step. Else, returns $\perp$.
- Searches $(X, *, *, ID_A, ID_B, m)$ in the list L_s, if such a record (X, y, V, ID_A, ID_B, m) is found, goes to the next step. Else, returns $\perp$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_A)^{h_3}$, if not, returns $\perp$.
- Else computes $w = \hat{e}(X, S_B)$ and $h_1 = y \oplus m||ID_A||V$.
- Searches $(*, h_1)$ in the list L_1, if such a record is found, the first element defined to be w and returns m. Else, returns $\perp$.

Now we discuss the difference between Exp 1 and Exp 2. The adversary can distinguish Exp 1 from Exp 2 if the following events happened. Firstly, during the Signcrypt query, if the adversary has made the query $H_1(w)$, where w happened to be the vacant value of some record. The probability of such event happening is at most $q_1/2^{n_4}$. The adversary made q_s Signcrypt query. So the probability of such events happening is at most $(q_1 \cdot q_s)/2^{n_4}$ in total. Secondly, during the Unsigncrypt query, if the adversary has guessed plaintext of some ciphertext. The probability of such event happening is at most $1/(2^{n_2} \cdot 2^{n_1} \cdot 2^{n_3})$. The adversary made q_u Unsigncrypt query. So the probability of such events happening is at most $q_u/(2^{n_2} \cdot 2^{n_1} \cdot 2^{n_3})$ in total. Let S_1 denote adversary can attack successfully in Exp 2. So, we have:

$$|\Pr(S_0) - \Pr(S_1)| \leq (q_{h_1} \cdot q_s)/2^{n_4} + q_u/(2^{n_2} \cdot 2^{n_1} \cdot 2^{n_3}).$$

Appendix B. Proof of Theorem 4

Weak BCDH problem. $(G_1, +)$ and $(G_2, \cdot)$ are two cycle groups of prime order q, P is a generator of G_1, $\hat{e} : G_1 \times G_1 \to G_2$ is a bilinear map between G_1 and G_2. Given $(P, aP, bP, cP, \frac{1}{c}P)$, where $a, b, c \in Z_q^*$, the strong BDH problem is to compute $\hat{e}(P, P)^{abc}$.

Proof. If there is an IND-CCA2 adversary A of IDGSC in the signcryption mode, then the challenger C can use it to solve the strong BDH problem. Let $(P, aP, bP, cP, \frac{1}{c}P)$ be an instance of the weak BCDH problem that C wants to solve. At first, C runs the $Setup(1^k)$ algorithm to produce parameters $params$. It sets the public key as $P_{Pub} = cP$, although it does not know the master key c. And then C sends $(params, P_{Pub})$ to the adversary A.

Besides the four lists L_i, $i = 0, 1, 2, 3$, the challenger C also needs to keep another list L_s which is used to record answers to the Signcrypt query.

Phase 1

Simulator: $H_0(ID_U)$

At the beginning, C chooses i_b uniformly at random from $1, \ldots q_0$. We assume that A does not make repeat queries.

- If $i = i_b$ responds with $H_0(ID_U) = bP$ and sets $ID_U = ID_b$.
- Else chooses k uniformly at random from Z_q^*; computes $Q_U = kP$ and $S_U = kP_{Pub}$; stores (ID_U, Q_U, S_U, k) in L_0 and responds with Q_U.

Simulator: $H_1(w)$

- Searches (w, h_1) in List L_1. If such a pair is found, returns h_1.
- Otherwise chooses h_1 randomly from $\{0, 1\}^{n_2} \times \{0, 1\}^{n_2} \times G_1^*$, and puts (w, h_1) into L_1 and returns h_1.

Simulator: $H_2(m||ID_1||ID_2)$

- Searches $(m||ID_1||ID_2, h_2)$ in List L_2. If such a pair is found, returns h_2.
- Otherwise chooses h_2 randomly from Z_q^*, and puts $(m||ID_1||ID_2, h_2)$ into L_2 and returns h_2.

Simulator: $H_3(m||X)$

- Searches $(m||X, h_3)$ in the list L_3. If such a pair is found, returns h_3.
- Otherwise chooses h_3 randomly from Z_q^*, and puts $(m||X, h_3)$ into L_3 and returns h_3.

Simulator: *Extract* (ID_U)
We assume that A makes the query $H_0(ID_U)$ before it makes extract query for ID_U.

- If $ID_U = ID_b$, aborts the simulation.
- Else, searches L_0 for the entry (ID_U, Q_U, S_U, k) corresponding to ID_U, and responds with S_U.

Simulator: *Sign*(ID_1, m)
We assume that A makes the query $H_0(ID_1)$ before *Sign*(ID_1, m) query.

Case 1: $ID_1 \neq ID_b$

- Find the entry (ID_1, Q_1, S_1, k) in L_0.
- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||ID_1||0)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2 P + h_3 \cdot S_1)$.
- Outputs $(X, m||ID_1||V)$. (Here H_i, $i = 2, 3$, comes from the simulator above.)

Case 2: $ID_1 = ID_b$

- Selects r uniformly from Z_q^*, and computes $X = rP_{Pub}$.
- Computes $h_2 = H_2(m||ID_1||0)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2 \cdot \frac{1}{c} P + h_3 \cdot bP)$.
- Outputs $(X, m||ID_1||V)$. (Here H_i, $i = 2, 3$, comes from the simulator above.)

Simulator: *Verify*(ID_1, σ)

- Computes $h_2 = H_2(m||ID_1||0)$, If $(m||ID_1||0, h_2) \notin L_2$, returns $\perp$.
- Computes $h_3 = H_3(m||X)$, If $(m||X, h_3) \notin L_3$, returns $\perp$.
- If $ID_1 \notin L_0$, returns $\perp$; else computes $Q_0 = H_0(ID_1)$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_1)^{h_3}$, if not, returns $\perp$. Else, returns $\top$.

Simulator: *Encrypt* (ID_2, m)
We assume that A has made the $H_0(ID_2)$ query before *Encrypt* (ID_2, m) query.

- Selects r uniformly from Z_q^*, and computes $X = rP$.
- Computes $h_2 = H_2(m||0||ID_2)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1} h_2 P$.
- Computes $Q_B = H_0(ID_2)$ and $w = \hat{e}(P_{Pub}, Q_2)^r$.
- Computes $h_1 = H_1(w)$ and $y = m||0||V \oplus h_1$.
- Outputs (X, y). (Here H_i, $i = 0, 1, 2, 3$, comes from the simulator above.)

Simulator: *Decrypt* (ID_2, ε)
We assume that A makes the query $H_0(ID_2)$ before *Decrypt* (ID_2, ε).

Case 1: $ID_2 \neq ID_b$

- Find the entry (ID_2, Q_2, S_2, k) in L_0.
- Computes $w = \hat{e}(X, S_2)$ and $h_1 = H_1(w)$.
- If $(w, h_1) \notin L_1$, returns $\perp$. Else, computes $m||0||V = y \oplus h_1$.
- Computes $h_2 = H_2(m||0||ID_2)$, If $(m||0||ID_2, h_2) \notin L_2$, returns $\perp$.
- Computes $h_3 = H_3(m||X)$, If $(m||X, h_3) \notin L_3$, returns $\perp$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2}$, if not, returns $\perp$. Else, returns m.

Case 2: $ID_2 = ID_b$
Step through the list L_1 with entries (w, h_1) as follows:

- Computes $m||0||V = y \oplus h_1$.
- If $m||0||ID_2 \in L_2$, computes $h_2 = H_2(m||0||ID_2)$; else moves to the next entry in L_1 and begin again.
- If $m||X \in L_3$, computes $h_3 = H_3(m||X)$; else moves to the next entry in L_1 and begin again.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2}$. If so, returns m; else moves to the next entry in L_1 and begin again.
- If no message has been returned after stepping through L_1, return $\perp$.

Simulator: *Signcrypt* (ID_1, ID_2, m)
We assume that A makes the query $H_0(ID_1)$ and $H_0(ID_2)$ before making signcrypt query using identity ID_1 and ID_2.

Case 1: $ID_1 \neq ID_b$

- Find the entry (ID_1, Q_1, S_1, k) in L_0.
- Selects r uniformly from Z_q^*, and computes $X = rP$.

- Computes $h_2 = H_2(m||ID_1||ID_2)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2P + h_3S_1)$.
- Computes $Q_2 = H_0(ID_2)$ and $w = \hat{e}(P_{Pub}, Q_2)^r$.
- Computes $h_1 = H_1(w)$ and $y = m||ID_1||V \oplus h_1$.
- Outputs (X, y). (Here H_i, $i = 0, 1, 2, 3$, comes from the simulator above.)

Case 2: $ID_1 = ID_b$

- Find the entry (ID_2, Q_2, S_2, k) in L_0.
- Selects r uniformly from Z_q^*, and computes $X = rP_{Pub}$.
- Computes $h_2 = H_2(m||ID_1||ID_2)$ and $h_3 = H_3(m||X)$.
- Computes $V = r^{-1}(h_2 \cdot \frac{1}{c}P + h_3 \cdot bP)$.
- Computes $w = \hat{e}(X, S_2)$, $h_1 = H_1(w)$ and $y = m||ID_1||V \oplus h_1$.
- Outputs (X, y). (Here H_i, $i = 1, 2, 3$, comes from the simulator above.)

Simulator: *Unsigncrypt*$(ID_1, ID_2, \varepsilon)$

We assume that A makes the query $H_0(ID_1)$ and $H_0(ID_2)$ before making this query using these identities.

Case 1: $ID_2 \neq ID_b$

- Find the entry (ID_2, Q_2, S_2, k) in L_0.
- Computes $w = \hat{e}(X, S_2)$ and $h_1 = H_1(w)$.
- If $(w, h_1) \notin L_1$, returns $\perp$. Else, computes $m||ID_1||V = y \oplus h_1$.
- Computes $h_2 = H_2(m||ID_1||ID_2)$, If $(m||ID_1||ID_2, h_2) \notin L_2$, returns $\perp$.
- Computes $h_3 = H_3(m||X)$, If $(m||X, h_3) \notin L_3$, returns $\perp$.
- If $ID_1 = ID_2$ or $ID_1 \notin L_0$, returns $\perp$; else computes $Q_1 = H_0(ID_1)$.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_1)^{h_3}$, if not, returns $\perp$. Else, returns m.

Case 2: $ID_2 = ID_b$

Step through the list L_1 with entries (w, h_1) as follows:

- Computes $m||ID_1||V = y \oplus h_1$.
- If $ID_1 = ID_2$ or $ID_1 \notin L_0$, moves to the next entry in L_1 and begin again; else computes $Q_1 = H_0(ID_1)$.
- If $m||ID_1||ID_2 \in L_2$, computes $h_2 = H_2(m||ID_1||ID_2)$; else moves to the next entry in L_1 and begin again.
- If $m||X \in L_3$, computes $h_3 = H_3(m||X)$; else moves to the next entry in L_1 and begin again.
- Checks that $\hat{e}(X, V) = \hat{e}(P, P)^{h_2} \cdot \hat{e}(P_{Pub}, Q_1)^{h_3}$. If so, returns m; else moves to the next entry in L_1 and begin again.
- If no message has been returned after stepping through L_1, return $\perp$.

Challenge. At the end of Phase 1, the adversary A outputs two identities, ID_A and ID_B, two messages, m_1 and m_2. If $ID_B \neq ID_b$, aborts the simulation; else it sets $X^* = aP$ and then chooses $\gamma \in \{0, 1\}$, and $y^* \in \{0, 1\}^{n_2} \times \{0, 1\}^{n_2} \times G_1^*$ at random. At last, it returns the challenge ciphertext $\delta^* = (X^*, y^*)$ to A.

Phase 2.

The queries made in Phase 2 are responded in the same way as those made in Phase 1. Here, the queries follow the restrictions that are defined in Game 6.

Guess.

At the end of Phase 2, A outputs a bit γ'. If $\gamma' = \gamma$, the challenger C outputs the answer to the weak BCDH problem:

$$w^* = \hat{e}(X^*, S_B) = \hat{e}(P, P)^{abc}.$$

Let's analyze the probability that the simulation can succeed. Two simulators need to be noted. First, in the challenge stage, the simulator hopes that the adversary chosen ID_b as the target recipient identity. This will be the case with probability at least $1/q_0$. If this is not the case, there will be an error when the adversary tries to make query *Extract*(ID_b). Second, in Phase 2, if the adversary makes query $H_1(w = \hat{e}(P, P)^{abc})$, the simulation will fail. However, with probability $1/q_1$ the challenger can guess the answer of the weak BCDH problem from the records in List L_1. From the above remarks we conclude that the challenger can solve the weak BCDH problem with probability at least: $Adv^{ind-cca2}_{A^{idgsc-in-sc}}(t, p)/(q_0 \cdot q_1)$.

References

[1] D. Boneh, M. Franklin, Identity based encryption from the Weil pairing, in: Advances in Cryptology- Crypto'01, in: LNCS, vol. 2139, Springer, 2001.
[2] F. Bao, R.H. Deng, A signcryption scheme with signature directly verifiable by public key, in: Proceeding of PKC'98, in: LNCS, vol. 1431, Springer-Verlag, 1998, pp. 55–59.
[3] X. Boyen, Multipurpose identity-based signcryption: a Swiss army knife for identity-based cryptography, in: D. Boneh (Ed.), Advances in Cryptology-CRYPT0 2003, in: Lecture Notes in Computer Science, vol. 2729, Springer-Verlag, Berlin, 2003, pp. 383–399.
[4] R. Canetti, S. Halevi, J. Katz, Chosen-ciphertext security from identity-based encryption, in: Advances in Cryptology-EUROCRYPT 2004, in: LNCS, vol. 3027, Springer-Verlag, 2004, pp. 207–222.
[5] L. Chen, Malone-Lee, Improved identity-based signcryption, in: S. Vaudenay (Ed.), Public Key Cryptography-PKC2005, in: Lecture Notes in Computer Science, vol. 3386, Springer-Verlag, Berlin, 2005, pp. 362–379.
[6] Certicom Research, Standards for efficient cryptography, SEC 1: elliptic curve cryptography, Standards for efficient cryptography group (SECG), September 20, 2000.

[7] S.S.M. Chow, S.M. Yiu, L.C.K. Hui, K.P. Chow, Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity, in: J.I. Lim, D.H. Lee (Eds.), Information Security and Cryptology-ICISC'03, in: Lecture Notes in Computer Science, vol. 2971, Springer-Verlag, Berlin, 2004, pp. 352–369.
[8] R. Hwang, C. Lai, F. Su, An efficient signcryption scheme with forward secrecy based on elliptic curve, Applied Mathematics and computation 167 (2005) 870–881.
[9] Y. Han, X. Yang, New ECDSA-verifiable generalized signcryption, Chinese Journal of Computer 11 (2006) 2003–2012.
[10] H.Y. Jung, K.S. Chang, D.H. Lee, J.I. Lim, Signcryption schemes with forward secrecy, proceesing of WISA 2 (2001) 403–475.
[11] S. Lal, P. Kushwah, ID based generalized signcryption, Cryptology Eprint Archive, 2008/084.
[12] B. Libert, J. Quisquater, A new identity based signcryption schemes from pairings, in: Proceeding of the 2003 IEEE Information Theory Workshop, Paris, France, 2003, pp. 155–158.
[13] Malone-Lee, Identity Based Signcryption. Cryptology ePrint Archive, Report 2002/098.
[14] K.G. Paterson, ID-based signatures from pairings on elliptic curves, Electroniics Letters 38 (18) (2002) 1025–1026.
[15] A. Shamir, Identity-based cryptosystems and signature schemes, in: G.R. Blakley, D. Chaum (Eds.), Advances in Cryptology-CRYPT0'84, in: Lecture Notes in Computer Science, vol. 196, Springer-Verlag, Berlin, 1984, pp. 47–53.
[16] X. Wang, X. Yang, Y. Han, Provable secure generalized signcryption. Cryptology Eprint Archive, 2007/173.
[17] Y. Zheng, Digital signcryption or How to Achieve Cost (Signature Encryption) $\leq$ Cost (Signature) + Cost (Encryption), in: CRYPTO'97, in: LNCS, vol. 1294, Springer-Verlag, Berlin, 1997, pp. 165–179.
[18] Y. Zheng, H. Imai, How to construct efficient signcryption schemes on elliptic curves, Information Processing Letters 68 (5) (1998) 227–233.